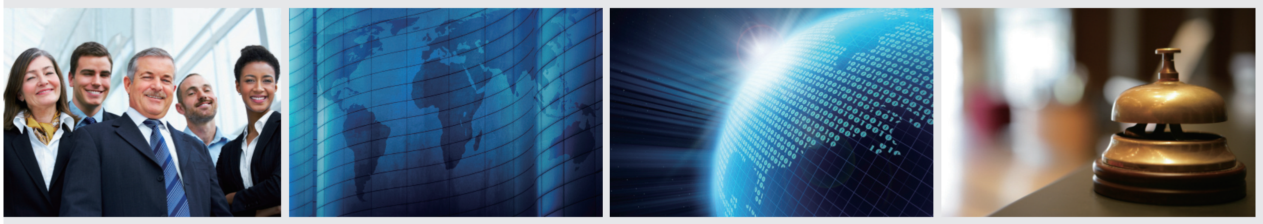
Lanner

creating value in applied computing

Embedded & Industrial Computing

Hardware Platforms for Embedded and Industrial Computing



LEC-7105
Version 1.0

User's Manual

Publication date:2012-01-03

About

Overview

Icon Descriptions

The icons are used in the manual to serve as an indication of interest topics or important messages. Below is a description of these icons:

NOTE: This check mark indicates that there is a note of interest and is something that you should pay special attention to while using the product.

WARNING: This exclamation point indicates that there is a caution or warning and it is something that could damage your property or product.

Online Resources

The listed websites are links to the on-line product information and technical support.

Resource	Website
Lanner	http://www.lannerinc.com
Product Resources	http://assist.lannerinc.com
RMA	http://eRMA.lannerinc.com

Copyright and Trademarks

Acknowledgement

Intel, Pentium and Celeron are registered trademarks of Intel Corp.

Microsoft Windows and MS-DOS are registered trademarks of Microsoft Corp.

All other product names or trademarks are properties of their respective owners.

Compliances and Certification

CE Certification

This product has passed the CE test for environmental specifications. Test conditions for passing included the equipment being operated within an industrial enclosure. In order to protect the product from being damaged by ESD (Electrostatic Discharge) and EMI leakage, we strongly recommend the use of CE-compliant industrial enclosure products.

FCC Class A Certification

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Table of Contents

Chapter 1: Introduction

Thank you for choosing the LEC-7105. The LEC-7105 is Lanner's flagship IPC. It features the Dual Core Intel® Atom™ D525 processor that has 1.8GHz of processing power.

The LEC-7105 is an ideal solution for digital signage and public infortainment. All electronics are protected in a compact sealed aluminum case as a stand-alone unit and can be easily situated in a place where space is limited and the weather condition is diverse.

A solid sealed Aluminum extrusion framing provides vibration and dust resistance while providing a passive cooling solution. It also provides great protection from EMI and shock.

Here is the list of the key features:

- Intel integrated Graphics Media Accelerator 3150 which supports VGA (up to 2048x1536) and DVI-D (1920 x1080)
- Dual 10/100/1000 Mbps LAN (support WOL (Wake-on-LAN) and Remote-wake-up)
- Two Mini-PCIe expansion slots (One Mini-PCIe comes with a SIM card reader that can support 3G Internet and the other Mini-PCIe can support Wi-Fi or Bluetooth connection)
- One power eSATA (5V external SATA) which also supports USB connectivity. The Power eSATA solution incorporates the eSATA connector with power source together, allowing you to use external SATA devices without the need of additional power source. It provides storage for photos, videos and other multi-media contents.
- USB x 4, COM x 2 (COM1 is RS-232 and COM2 is RS-232/422/485 selectable, and Digital Input/Output (through 2 x 5-pin terminal block)
- Audio output for L/R channels with RCA connectors (Realtek ALC888S codec)

System Specification

LEC 7 Series		LEC-7105
Dimension (WxHxD)		268x44x174mm (10.55"x1.73"x6.85")
Processor		Intel Atom D525 1.8GHz
Chipset		Intel ICH8M
System Memory	Technology	DDR3 SODIMM x1
	Max. Capacity	Up to 4GB
Storage	IDE	CF socket Type I/II x1
	SATA	2.5" HDD/SSD drive bay x1
Ethernet Controller		Realtek RTL8111 x2
Graphic Controller		Intel GMA3150
Audio Controller		Realtek ALC888S
IO	LAN	GbE RJ45 x2
	Display	DB15 x1 for VGA, DVI-D (up to1920x1080)
	Video Grabber	No
	Audio	RCA x2 for right/left Line-out channels, Internal pin header for Line-out, Line-in and Mic-in
	Serial I/O	DB9 x2 for RS232 x1; RS232/422/485 x1
	GPS	No
	Digital I/O	2 x 5-pin terminal block for DI x4 and DO x4 (5V TTL)
	USB 2.0	Type A x4; Internal x2
	Power Input	DC jack with lock
	Expansion	Mini-PCIe x2 (one with SIM card reader)
	Others	External: Power-on button, Power-on switch, 3x SMA antenna hole, reset. Internal: PS/2 keyboard and mouse, +5Vdc output
Power Input		+12Vdc +/- 5%, ATX mode
AC Adapter		60W +12V @ 5A
Hardware Monitor		Winbond W83627UHG integrated watchdog timer 1~255 level
OS Support		Linux , XPE/WES2009, XP PRO FES, WS7E, WS7P, WIN 7 PRO-E
Certifications		CE, FCC Class A
Compliance		No
Operating Temperature Range with Commercial Components		-5~45°C/23~113°F

Chapter 1

Introduction

Package Contents

Your package contains the following items:

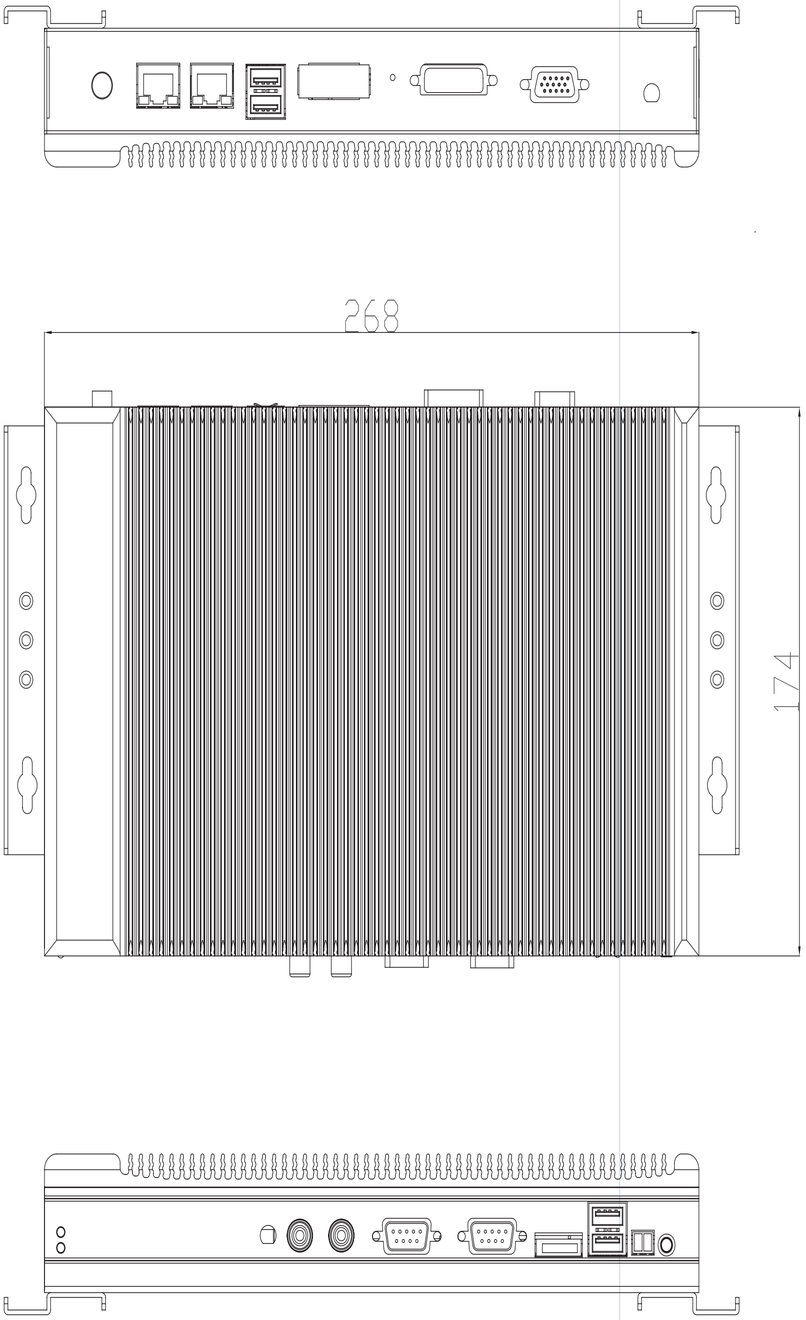
- LEC-7105 Embedded System
- DC+12V 60W Power Adapter (080W240318306, US type)
- Serial-ATA/Power Cable (P/N: 080W1N2201001)
- Wall-Mounting Kit (P/N: SE9ESA900R100)
- Drivers and User's Manual CD

Chapter 2: System Components

System Drawing

Mechanical dimensions of the LEC-7105

Unit: mm

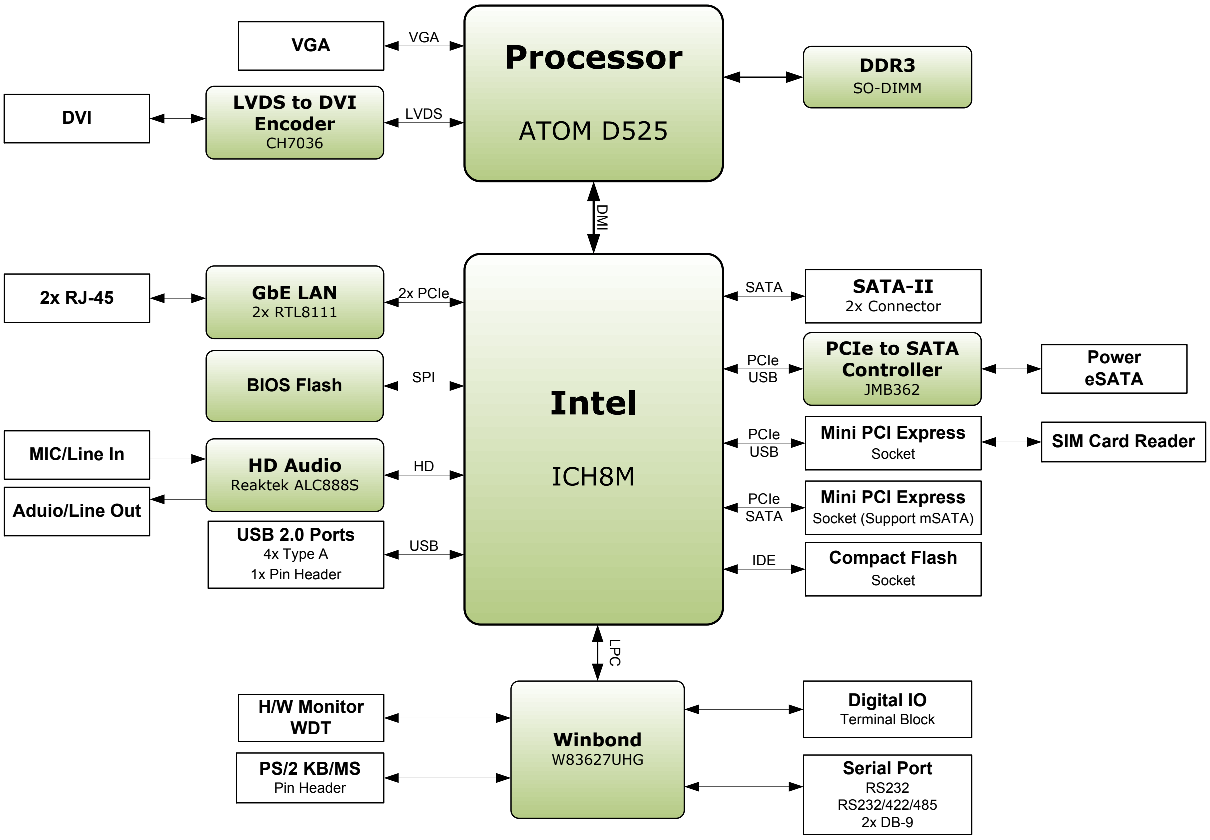


Chapter 2

System Components

Block Diagram

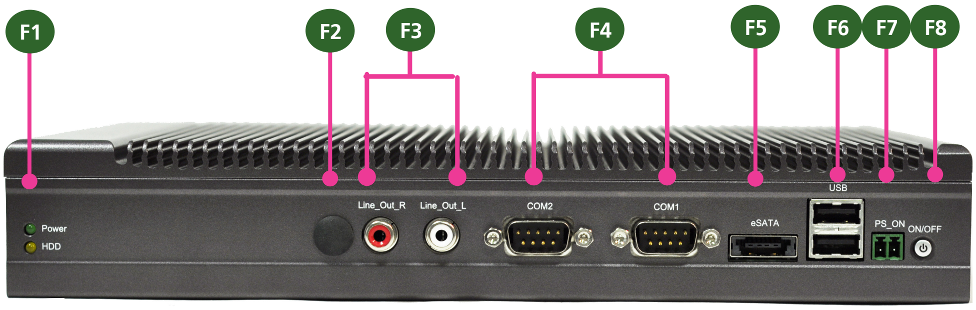
The block diagram depicts the relationships among the interfaces and modules on the motherboard..



Chapter 2

System Components

Front Components

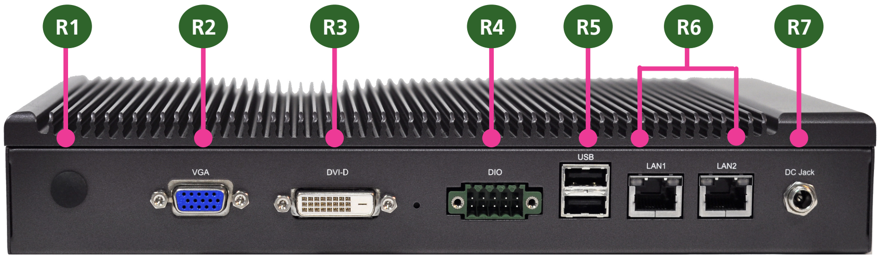


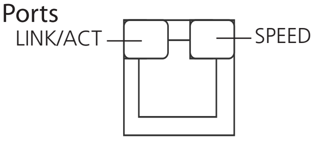
Component	Description	Pin Definition Reference
F1 HDD (Yellow) and Power LED (Green)	HDD • Blinking: data access activities • Off: no data access activities Power • On: The computer is on. • Off: The computer is off .	
F2 Antenna Hole	Reserved for antenna	
F3 Line_Out_R Line_Out_L	RCA Jack for audio output left and right	CN1, CN2 on page 17
F4 Serial Ports 1 and 2	Serial ports through the DB-9 connector; COM1 supports RS-232 and COM2 supports RS-232/422/485 with switch selection among RS-232/422/485.	COM1, COM2 on page 14
F5 Power eSATA	An external SATA connector with 5V power supply and support hot plugging. It also supports USB 2.0 connection.	EUSB1 on page 14
F6 Dual USB Stack Connector	An USB type A connector; in addition to this connector, an internal pin header is provided.	Dual USB Port Connectors (USB1, USB2) on Page16
F7 Power-on Switch	A power-on switch through the Phoenix contact for distant power-on/off control	J12 on page 16
F8 Power Button with dual LED	ATX Power-on button with LEDs: Standby mode in Red; Power-on mode in Green	

Chapter 2

System Components

Rear Components

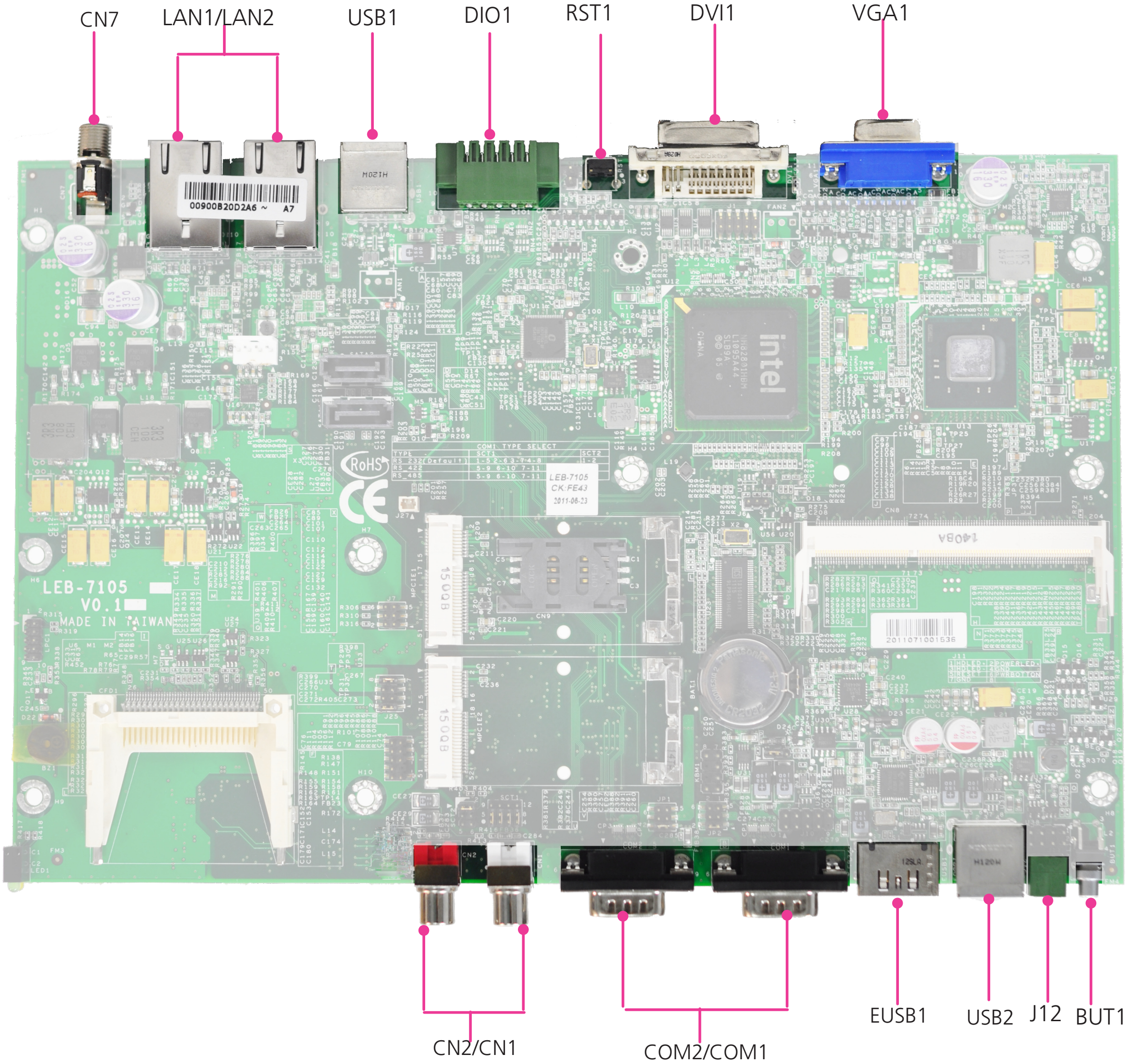


Component	Description	Pin Definition Reference
R1 Antenna Hole	Reserved for antenna	
R2 VGA Port	DB-15 Female Connector for VGA connection (up to 2048x1536)	VGA1 on page 17
R3 DVI-D	DVI-D port (single link) is provided by Intel GMA 3150 through the Chrontel's CH7036 LVDS to DVI converter.	DVI1 Connector on page 17
R4 DIO Port	4 digital input and 4 output ports to support input and output operations.	DIO1 on page 15
R5 Dual USB Stack Connector	An USB type A connector; in addition to this connector, an internal pin header is provided.	Dual USB Port Connectors (USB1, USB2) on Page 16
R6 Dual 10/100/1000 LAN Ports 	Two RJ-45 (network) jacks with LED indicators as described below. The LAN ports are provided by Realtek RTL8111. They both support WOL (Wake-on-LAN) and Remote-wake-up. LINK/ACT (Yellow) • On/Flashing: The port is linking and active in data transmission. • Off: The port is not linking. SPEED (Green/Amber) • Amber: The connection speed is 1000Mbps. • Green: The connection speed is 100Mbps • Off: .The connection speed is 10Mbps.	LAN Ports (LAN1/LAN2) on page 15
R7 DC Jack	DC-in 12V power socket with Lock. Only use the power adapter supplied with the LEC-7105 System.	

Chapter 3:
Board Layout

External Connectors

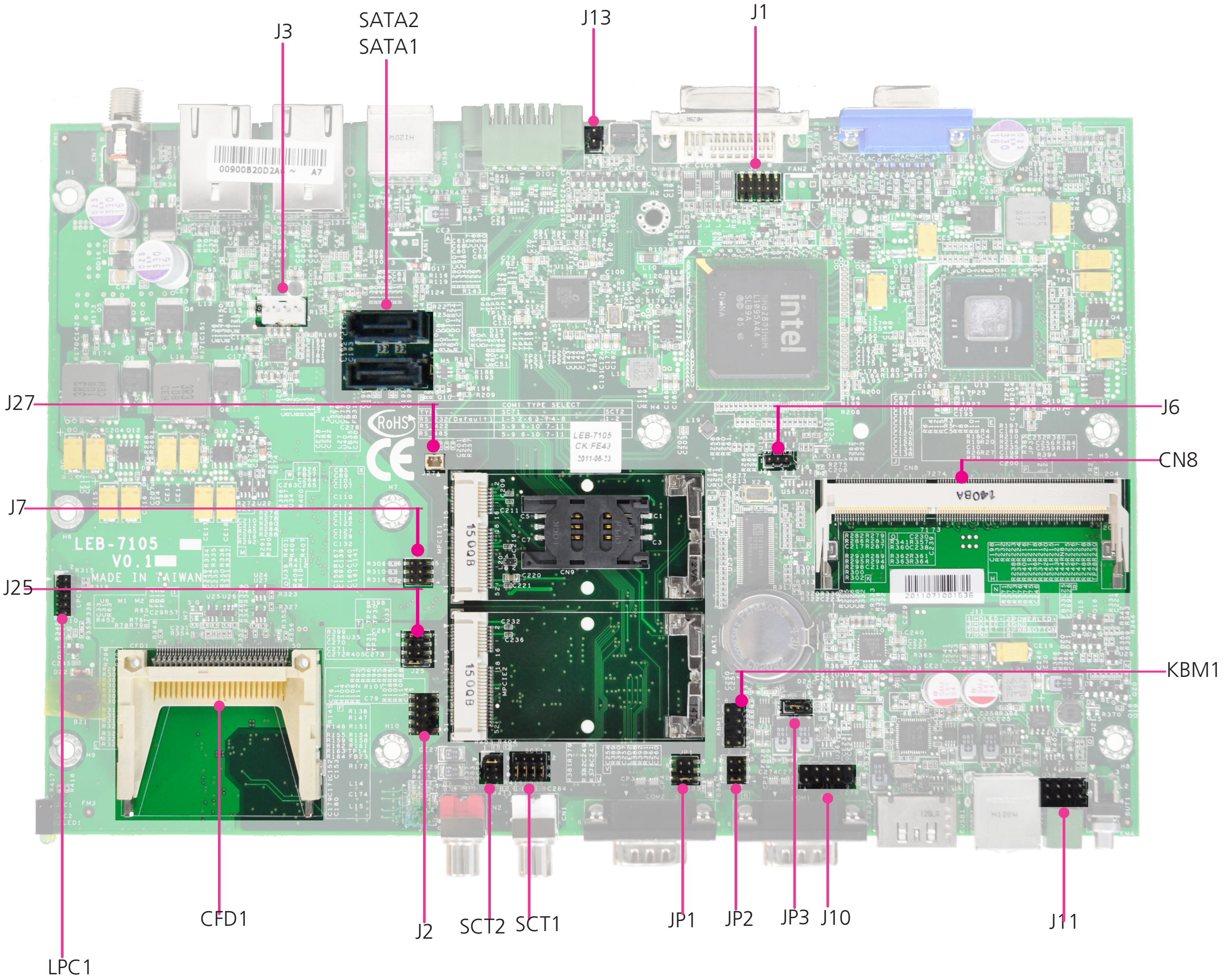
The following picture highlights the location of system input/output connectors. Refer to the table 3.1 Connector List for more details.



Chapter 3

Internal Connectors and Jumpers

The following picture highlights the location of internal connectors and jumpers. Refer to the table 3.1 Connector List for more details.



Chapter 3

Connectors and Jumpers List

The tables below list the function of each of the board jumpers and connectors by labels shown in the above section. The next section in this chapter gives pin definitions and instructions on setting jumpers.

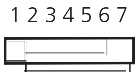
Table 3.1 Connector List for **LEB-7105**

Labels	**Function**	**Pin Definition Reference Page**
CFD1	CompactFlash	P15
CN1 & CN2	Lineout Left/Right	P17
COM1	RS-232 Port	P14
COM2	RS-232/422/485 Port	P14
DIO1	Digital Input/Output	P15
DVI1	DVI-D Connector	P17
EUSB1	Power eSATA Port	P14
J1	SPI ROM Header	Reserved for Factory
J10	USB Pin Header	P16
J11	Miscellaneous Front Panel Pins	P16
J12	External Power Switch	P16
J13	SYSTEM Thermal Sensor	P17
J2	Line In/Out and MIC Pin Header	P17
J25	Mini-PCIe Power Voltage Selection	P17
J27	Power for Passive Antenna	P18
J3	SATA Power	P14
J6	ICH8M Chipset SMB Signals	Reserved for Factory
J7	LAN and WLAN LED (Only on MPCIE1)	P17
JP1 & JP2	Select COM1/COM2 Pin9 Function Jumper Settings	P14
JP3	Clean CMOS	P15
KBM1	PS/2 Keyboard and Mouse	P17
LAN1/LAN2 Ports	LAN1, LAN2 ports	P15
LPC1	Low Pin Count Bus for Debug Purpose	Reserved for Factory
MPCIE1	Mini-PCIe Slot (with SIM Card Reader)	P16
MPCIE2	Mini-PCIe Slot	P16
SATA1	Serial-ATA Connector 1	P14
SATA2	Serial-ATA Connector 2	P14
SCT1/SCT2	Seclect COM2 Protocol Jumper settings	P14
USB1	Dual USB Port	P16
USB2	Dual USB Port	P16
VGA	DB-15 VGA Port	P17

Jumper Settings

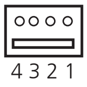
LEB-7105

Serial-ATA Connector (SATA1, SATA2): It is for connecting a 2.5" harddisk to be served as your system's storage. It can support SATA II which features Data transfer rates up to 3.0 Gb/s (300 MB/s).



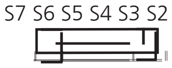
Pin No.	Function
1	GND
2	TX0_+
3	TX0_-
4	GND
5	RX0_-
6	RX0_+
7	GND

4-pin Serial-ATA Power Connector (J3): It is for connecting the SATA power cord.



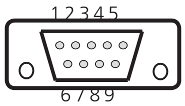
Pin No.	Function
1	+5V
2	GND
3	GND
4	+12V

Power eSATA Port (5V, EUSB1): A Power external SATA port supports hot plugging of SATA II disc. It was provided by the PCIe to SATA controller: JMB362 which connects to the ICH8M through the PCIe interface. It can support USB2.0 as well as eSATA transmission.



Pin No.	Function	Pin No.	Function
1	GND	1	+5V
2	TX1_+	2	USB8+
3	TX1_-	3	USB8-
4	GND	4	GND
5	RX1_-		
6	RX1_+		
7	VCC5		

RS-232 Serial Port (COM1): It is a RS-232 port through the D-SUB9 connector.

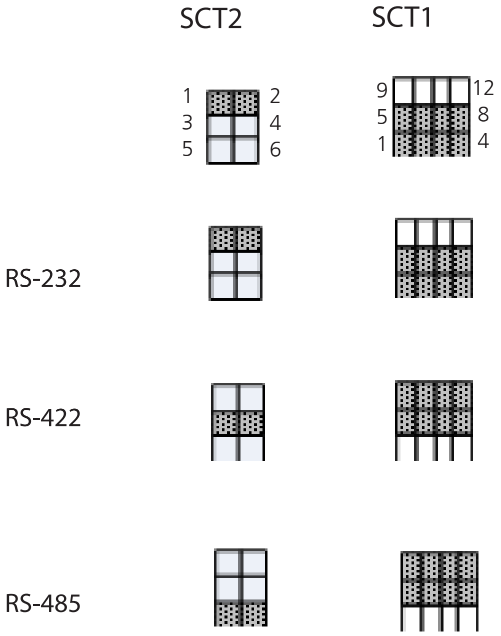


Pin No.	Pin Name	Pin No.	Pin Name
1	DCD	6	DSR
2	RXD	7	RTS
3	TXD	8	CTS
4	DTR	9	RIA
5	GND		

RS-232/422/485 Serial Port (COM2): It is a RS-232/422/485 port through the D-SUB9 connector.

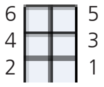
Pin No.	Pin Name		
	RS-232	RS-422	RS-485
1	DCD	TxD-	Data-
2	RXD	TxD+	Data+
3	TXD	RxD-	
4	DTR	RxD+	
5	GND		
6	DSR		
7	RTS		
8	CTS		
9	RI		

SCT1, SCT2: Select COM2 Protocol Setting



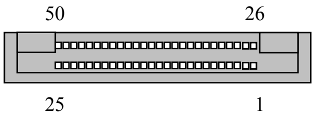
COM1 TYPE	SCT2	SCT1
RS-232 (Default)	1-2	1-5, 2-6, 3-7, 4-8
RS-422	3-4	5-9, 6-10, 7-11, 8-12
RS-485	5-6	5-9,6-10,7-11,8-12

JP1, JP2: Select COM1 and COM2 power : The Pin No. 9 of RS-232 can be altered to supply power. JP1 and JP2 are used to select the power voltage for COM1 and COM2 respectively.



RS-232 Pin 9 Function	JP1, JP2
+5V	1-2
+12V	3-4
RI (Default)	5-6

CompactFlash Connector (CFD1): It is for connecting a Compact Flash card to be served as your system's storage.



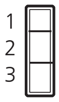
Pin No.	Function	Pin No.	Function
Pin	Signal	Pin	Signal
1	GND	26	CD1#
2	PDD3	27	PDD11
3	PDD4	28	PDD12
4	PDD5	29	PDD13
5	PDD6	30	PDD14
6	PDD7	31	PDD15
7	PDCS1_N	32	PDCS3_N
8	GND	33	N/A
9	GND	34	PDIOR_N
10	GND	35	PDIOW_N
11	GND	36	WE#
12	1GND	37	IRQ14
13	VCC5	38	VCC5
14	GND	39	CSEL#
15	GND	40	N/A
16	GND	41	PRST
17	GND	42	PDIORDY
18	PDA2	43	PDDREQ
19	PDA1	44	PDDACK
20	PDA0	45	PDACTIVE
21	PDD0	46	PATADET
22	PDD1	47	PDD8
23	PDD2	48	PDD9
24	IOCS16#	49	PDD10
25	CD2#	50	GND

LAN1/LAN2 Ports (LAN1/LAN2): The LAN ports are provided by Realtek RTL8111E Ethernet Controllers. The following lists its main features:

- Wake-on-LAN and remote wake-up support
- Microsoft NDIS5, NDIS6 Checksum Offload (IPv4, IPv6, TCP, UDP) and Segmentation Task-offload (Large send v1 and Large send v2) support
- Supports IEEE 802.1P Layer 2 Priority Encoding
- Supports IEEE 802.1Q VLAN tagging

Pin No.	Description	
	Fast Ethernet	Gigabit Ethernet
1	TX+	BI_DA+
2	TX-	BI_DA-
3	RX+	BI_DB+
4	--	BI_DC+
5	--	BI_DC-
6	RX-	BI_DB-
7	--	BI_DD+
8	--	BI_DD-

Clear CMOS jumper (JP3): It is for clearing the CMOS memory.



Pin No.	Pin Name
1-2	Normal (Default)
2-3	Clear CMOS

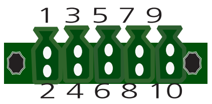
Digital I/O (DIO1)

Digital IN/OUT(DIO1) Connector: The 8 pins of digital Input/Output (GPIO) support input and output operations through the 2x5-pin terminal block.

TTL Level is +5V; Maximum input/output current for each port is 20mA			
Input/Output	Voltage	Logic	Register
	0~2V	Low	0
	2~5V	High	1
The output default value is 0			

DIO Address LDN8	
Address	Description
0x2e	SUPERIO_INDEX
0x2f	SUPERIO_DATA
0x07	BANK_REG
0xE6 (Bit 3)	GPO63 0: Low 1: High
0xE6 (Bit 2)	GPO62 0: Low 1: High
0xE6 (Bit 1)	GPO61 0: Low 1: High
0xE6 (Bit 0)	GPO60 0: Low 1: High

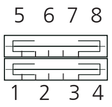
DIO Address LDN9	
Address	Description
0x2e	SUPERIO_INDEX
0x2f	SUPERIO_DATA
0x07	BANK_REG
0xE6 (Bit 3)	GPI24 0: Low 1: High
0xE6 (Bit 2)	GPI25 0: Low 1: High
0xE6 (Bit 1)	GPI26 0: Low 1: High
0xE6 (Bit 0)	GPI27 0: Low 1:High



Pin No.	Pin Name	Pin No.	Pin Name
1	Input0	2	Output0
3	Input1	4	Output1
5	Input2	6	Output2
7	Input3	8	Output3
9	GND	10	GND

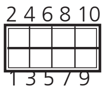
Dual USB Port Connector #0 and #1 (USB1):

Dual USB Port Connector #2 and #3 (USB2)



Pin No.	Pin Name
1	+5V
2	USBD1-
3	USBD1+
4	GND
5	+5V
6	USBD0-
7	USBD0+
8	GND

USB 2.0 Pin Header (J10, USB#4 and #5):

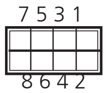


Pin No.	Pin Name	Pin No.	Pin Name
1	+5V	2	+5V
3	USBD4-	4	USBD5-
5	USBD4+	6	USBD5+
7	Ground	8	Ground
		10	NC

External Power Button (J12): The external power button is provided for distant power-on control.

PIN NO.	DESCRIPTION
1	PWR_BTN_N
2	GND

Front Panel Function Pin Header (J11): It provides redundant LED signal and button function on the front panel.



Pin No.	Pin Name	Function	Pin No.	Pin Name	Function
1	POWER_LED	HDD LED	2	PWR_LED+	Power LED
3	HD_LED		4	GND	
5	Reset	System Reset Button	6	POWER_BTN-	Power On/Off Push Button
7	GND		8	GND	

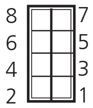
Mini PCI Express Connector 1(MPCIE1):

PIN	Pin Name	PIN	Pin Name
1	WAKE#	2	VCC3.3
3	N/A	4	GND
5	N/A	6	VCC1.5
7	CLKREQ#	8	VREG_USIM
9	GND	10	UIM_DATA
11	CLK_PCIE_MINI_N1	12	UIM_CLK
13	CLK_PCIE_MINI_P1	14	UIM_RESET
15	GND	16	UIM_VPP
17	RSV	18	GND
19	RSV	20	RF_KILL_N1
21	GND	22	PLTRST
23	PCIE_RX_N2	24	PCIE1_P24
25	PCIE_RX_P2	26	GND
27	GND	28	VCC1.5
29	GND	30	SMBCLK
31	PCIE_TX_N2	32	SMBDATA
33	PCIE_TX_P2	34	GND
35	GND	36	USB_N6
37	GND	38	USB_P6
39	VCC3.3	40	GND
41	VCC3.3	42	LED1_WWAN
43	GND	44	LED1_WLAN
45	RSV	46	LED1_WPAN
47	RSV	48	VCC1.5
49	RSV	50	GND
51	RSV	52	VCC3.3

Mini PCI Express Connector 2 (MPCIE2)

PIN	Pin Name	PIN	Pin Name
1	WAKE#	2	VCC3.3
3	N/A	4	GND
5	N/A	6	VCC1.5
7	CLKREQ#	8	N/A
9	GND	10	N/A
11	CLK_PCIE_MINI_N2	12	N/A
13	CLK_PCIE_MINI_P2	14	N/A
15	GND	16	N/A
17	RSV	18	GND
19	RSV	20	RF_KILL_N2
21	GND	22	PLTRST
23	PCIE_RX_N4	24	PCIE2_P24
25	PCIE_RX_P4	26	GND
27	GND	28	VCC1.5
29	GND	30	SMBCLK
31	PCIE_TX_N4	32	SMBDATA
33	PCIE_TX_P4	34	GND
35	GND	36	USB_N7
37	GND	38	USB_P7
39	VCC3.3	40	GND
41	VCC3.3	42	N/A
43	GND	44	N/A
45	RSV	46	N/A
47	RSV	48	VCC1.5
49	RSV	50	GND
51	RSV	52	VCC3.3

Mini PCI Express (MPCIE1/MPCIE2) Power Setting in Pin 24 (J25):

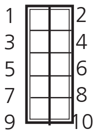


Connector	Description	J25
MPCIE1	+3.3V Standby (miniPCIe 1.2)	1-2
MPCIE1	+3.3V Default (miniPCIe 1.0)	5-6
MPCIE2	+3.3V Standby (miniPCIe 1.2)	3-4
MPCIE2	+3.3V Default (miniPCIe 1.0)	7-8

Line Out Left/Right (CN1/CN2)

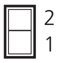
CN1		CN2	
Pin No.	Description	Pin No.	Description
1	GND	1	GND
2	FRONT_OUT_L	2	FRONT_OUT_R

Line In/Out and MIC Pin Header (J2)



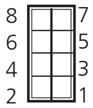
Pin No.	Description	Pin No.	Description
1	LINE_OUT2_R	2	LINE_PUT2_L
3	GND	4	GND
5	MIC_R	6	MIC_L
7	LINE_IN_R	8	N/A
9	LINE_IN_L	10	GND

SYSTEM Thermal Sensor (J13)



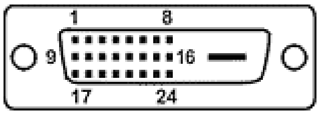
Pin No.	Description
1	SYS_TIN
2	GND

PS/2 Keyboard and Mouse (KBM1)



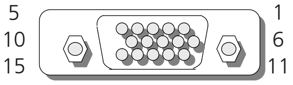
Pin No.	Description	Pin No.	Description
1	+5V	2	MCLK
3	MDATA	4	NC
5	KDATA	6	NC
7	GND	8	KCLK

DVI-D Connector (DVI1): A single link DVI-D connector



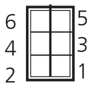
Pin No.	Description	Pin No.	Description
1	TXD_2-	9	TXD_1-
2	TXD_2+	10	TXD_1+
3	GND	11	GND
4	N/A	12	N/A
5	N/A	13	N/A
6	DDC_CLK	14	VCC5
7	DDC_DATA	15	GND
8		16	HPD
Pin No.	Description	Pin No.	Description
17	TXD_0-	C1	
18	TXD_0+	C2	
19	GND	C3	
20	NC	C4	
21	NC	C5	GND
22	GND	C6	GND
23	TXD_CLK_P		
24	TXD_CLK_N		

DB-15 VGA Connector (VGA1)



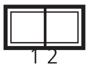
Pin No.	Description	Pin No.	Description
1	RED	6	CRT DET
2	GREEN	7	GND
3	BLUE	8	GND
4	N/A	9	VCC5
5	GND	10	GND
Pin No.	Description		
11	N/A		
12	DDC DAT		
13	HSYNC		
14	VSYNC		
15	DDC CLK		

LAN and WLAN LED (Only on MPCIE1, J7)



Pin No.	Description	Pin No.	Description
1	LED1_WWAN	2	+3.3V
3	LED1_WLAN	4	+3.3V
5	LED1_WPAN	6	+3.3V

Power for Passive Antenna (J27)



Pin No.	Description
1	+3.3V
2	GND

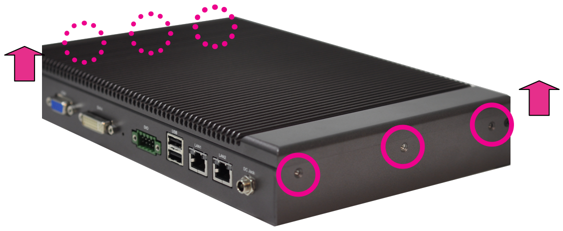
Chapter 4:
Hardware Setup

Preparing the Hardware Installation

To access some components and perform certain service procedures, you must perform the following procedures first.

WARNING: To reduce the risk of personal injury, electric shock, or damage to the equipment, remove the power cord to remove power from the server. The front panel Power On/Standby button does not completely shut off system power. Portions of the power supply and some internal circuitry remain active until AC power is removed.

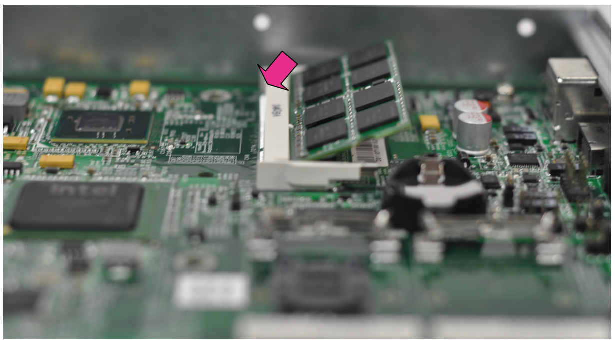
1. Unpower the LEC-7105 and remove the power cord.
2. Unscrew the 3 threaded screws on both sides of the top cover of the LEC-7105 System.
3. Slide the cover backwards and open the cover upwards.



Installing the System Memory

The motherboard supports DDR3 memory. It comes with one Double Data Rate (DDR3) Small Outline Dual Inline Memory Modules (SO-DIMM) sockets.

1. Open the SO-DIMM slot latches.
2. Install the SO-DIMM.



Note:

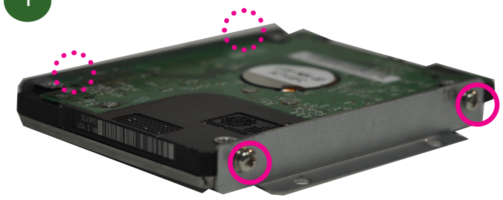
1. The motherboards can support up to 4 GB memory capacity in maximum.

Installing the Hard Disk

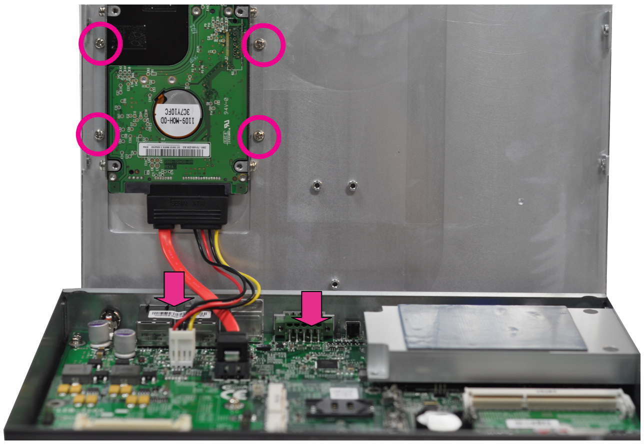
The system can accomdate two Serial-ATA disks. Follow these steps to install a hard disk into the LEC-7105:

1. Unsrew the 4 screws on the hard disk tray to take out the hard disk tray from the system.
2. Place hard disk on the hard disk tray and align the holes of the hard disk with the mounting holes on the tray.
3. Secure the hard disk with 4 mounting screws on the hard disk tray.
4. Connect the Serial-ATA power and datacables to the hard disk's connectors respectively.
5. Plug the Serial-ATA cable to the Serial-ATA Connector on the main board.
6. Put the hard disk tray with the installed hard disk back to the system and secure it with the mounting screws.

1



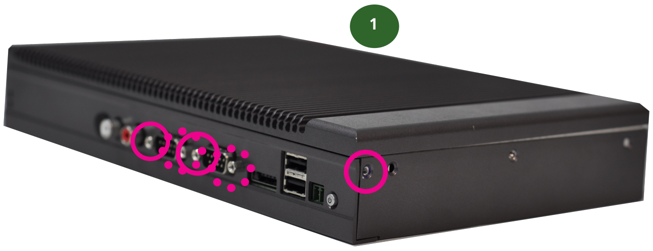
2

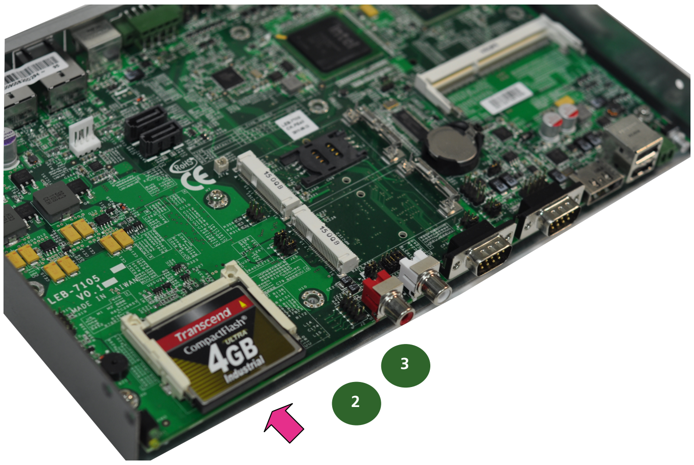


Installing a CompactFlash Card

LEC-7105 provides one CompactFlash slot. To install the CF card, Follow these procedures bellow for installing a CompactFlash card.

1. In order to insert the CF card, you will have to take off the front panel first. To take off the front panel, unscrew the 2 screws on the front panel and the hex-shaped screws of the COM ports.
2. Align CompactFlash card and the card slot with the arrow pointing toward the connector.
3. Push the card to insert into the connector.



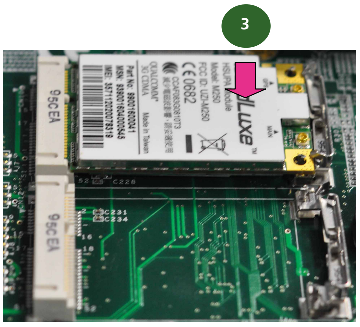


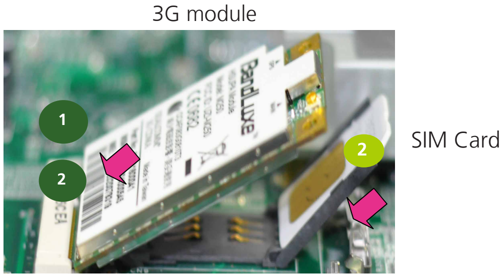
3G SIM Card Installation

1. Open the SIM tray and flip it diagnolly.
2. Align the cut corner of the SIM card with the SIM card socket. Make sure the ICs is in contact with the reader.
3. Insert the SIM card into the tray diagonally. Close and lock the tray.

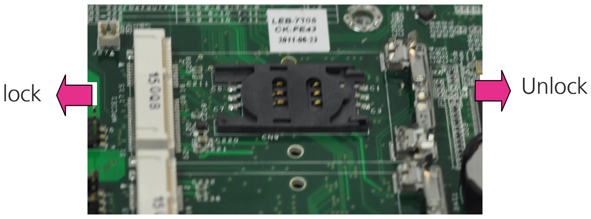
Wireless 3G module Installation

1. Align the wireless module's cutout with the Mini-PCIe slot notch.
2. Insert the wireless module into the connector diagonally.
3. Push the other end of the wireless module to be tightened with the latch.





Note:
1. To remove the module from the system, release the latch first by slightly bending it inward.
2. To remove the SIM card, unlock the tray first by sliding it outward.

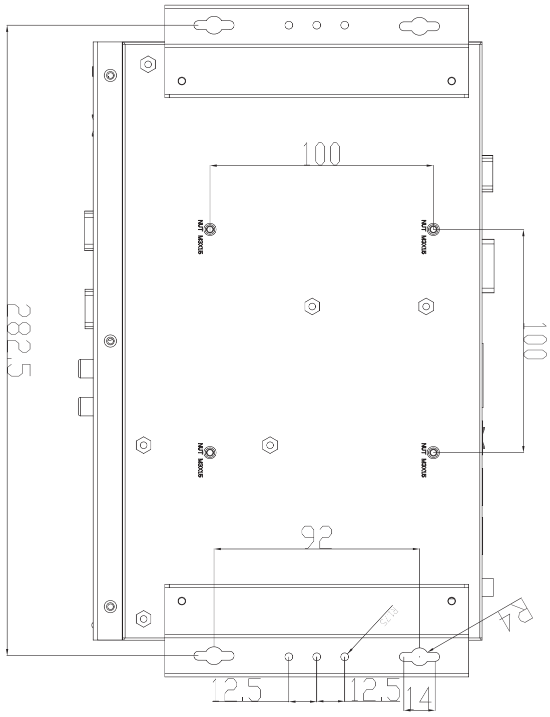


Wall Mounting

The product ships with wall mounting kit. To mount your product on the wall, follow the instructions below:

1. First make a hole for the anchor in the surface on the wall.
2. Then press the anchor into the hole until it is flush with the surface. You may need a hammer to tap the wall anchor.
3. Use a screwdriver to screw the threaded screw into the plastic anchor.
4. Attach the wall mounting bracket to the back of the device, securing it in place with four of the flat-head screws provided.
5. Hang the device on the wall.

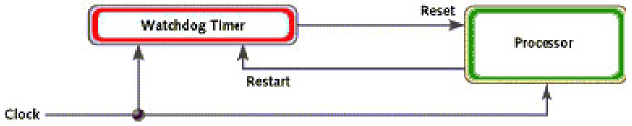
Unit: mm



Appendix A: Programming Watchdog Timer

A watchdog timer is a piece of hardware that can be used to automatically detect system anomalies and reset the system (or one pair of network ports in bypassed state; However, only one function can be activated at a time.) in case there are any problems. Generally speaking, a watchdog timer is based on a counter that counts down from an initial value to zero. The software selects the counter's initial value and periodically restarts it. Should the counter reach zero before the software restarts it, the software is presumed to be malfunctioning and the processor's reset signal is asserted. Thus, the processor will be restarted as if a human operator had cycled the power.

For sample watchdog code, see *wd_bp* folder under Driver and Utility on the *Driver and Manual CD*



Executing the commands through the Command Line:

1. wd_tst --swtsr (Set Watchdog Timeout State to Reset)
2. wd_tst --swt xxx (Set Watchdog Timer 1-255 seconds)
3. wd_tst[*] --start (Start Watchdog Timer)
4. wd_tst --stop (Stop Watchdog Timer)

The following procedures are required for running the watchdog program on DOS, Linux and FreeBSD.

Note:

1. For DOS environment, use DJGPP as compiler and the makefile: Makefile.dos.
2. For Linux, support kernel versions are 2.4.x and 2.6.x. Use the makefile:Makefile.linux.
3. For FreeBSD, support version is FreeBSD 8.0. Use the makefile: Makefile.

Build

To build program source code on Linux platform, use the following steps as a guideline:

1. Copy the proper makefile from the Driver and Manual CD to your system
2. Set the access mode with these two parameters by editing the Makefile.linux directly: DIRECT_IO_ACCESS= [0|1] (enter either 1 or 0) and LANNER_DRIVER= [0|1] (enter either 1 or 0). 1 is for direct access and no driver is needed. You will only need to execute the program directly. However, when it equaled to 0, driver installation is needed. Refer to the following Install section for more details.
3. Type make to build source code:

   make Makefile (Note: omit the file extensions)

After compiled, the executable program (bpwd_tst) and the driver (bpwd_drv.ko) will be in the bin subdirectory.

Install

The installation procedures depend on the access mode that you have set by using the above mentioned method.

If you have set DIRECT_IO_ACCESS=1, driver installation is not necessary. Proceed to the next section on executing

If you have set DIRECT_IO_ACCESS=0, Lanner bypass driver needs to be installed. Install the driver and create a node in the /dev directory as shown in the following example:

For Linux:

Insert module and create node in /dev as below example:

```
#insmod wd_drv.[k]o
#mknod /dev/wd_drv c 241 0
```

For FreeBSD:

Insert module as below example:

```
#kldload -v ./wd_drv.ko
```

Execute

```
# wd_tst --swtsb (Set Watchdog Timeout State to Bypass function)
# wd_tst --swtsr (Set Watchdog Timeout State to Reset function)
# wd_tst --swt xxx (Set Watchdog Timer 1-255 seconds)
# wd_tst[*] --start (Start Watchdog Timer)
# wd_tst --stop (Stop Watchdog Timer)
```

Note:

1. wd_tst --start will not be available if

DIRECT_IO_ACCESS=1, use the command: “./wd_tst --swt xxx” to start the watchdog timer instead .

2. Watchdog timer can support two functions, - system rest or LAN bypass. However, only one function can be activated at a time. You should modify the code or switch it to the desired state/function accordingly.
3. For more details, refer to the README file contained within the program.

A sample Watchdog program in C:

```
*********************************************************
**********************/
#include “../include/config.h”



#ifdef DJGPP

/* standard include file */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/* For DOS DJGPP */
#include <dos.h>
#include <inlines/pc.h>

#else //DJGPP
/* For Linux */


#ifdef DIRECT_IO_ACCESS
/* For Linux direct io access code */
/* standard include file */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#if defined(LINUX_ENV)
#include <sys/io.h>
#endif
#if defined(FreeBSD_ENV)
#include <machine/cpufunc.h>
#endif


#include <time.h>
#include <stdint.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#define delay(x) usleep(x)
#endif


#ifdef MODULE

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <asm/io.h>
#include <linux/delay.h>

#undef delay
#define delay(x) mdelay(x)
#undef fprintf
#define fprintf(S, A)  printk(A)

#endif //MODULE

#ifdef KLD_MODULE

#include <sys/types.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <sys/malloc.h>
#include <sys/kernel.h>
#include <sys/bus.h>
```

```
#include <sys/errno.h>


#include <machine/bus.h>
#include <machine/resource.h>


#endif


#endif


/* local include file */
#include "../include/ioaccess.h"


#if (defined(MODULE) || defined(DIRECT_IO_ACCESS) || defined(KLD_MODULE))



/*
 * Platform Depend GPIOs Interface for Watchdog and Lan bypass
 */
/*
 *---------------------------------------------------------------------------
 * LEB-7105 Version V1.0
 *
 * LEB-7105 embedded with HW Watchdog timer functions.
 * Set Lan bypass Enable/Disable while System-off:
 * ================================================
 * It is able to set Lan bypass enable/disable in system off mode by SW program.
 * The IO interface for off-mode bypass is connected to Winbond SIO 83627UHG
 * GPO22,GPO23(Pair1), GPO30,GPO31(Pair2),
 * Refer to Winbond 83527 datasheet for details.
 *
 * The truth table of function is defined as below:
 *
 *          Pair        Bypass function     GPIO Pin
 *    ---------------------------------------------------
 *          1           Enable              GPO22=1 GPO23=0
 *          1           Disable             GPO22=0 GPO23=1
 *          2           Enable              GPO30=1 GPO31=0
 *          2           Disable             GPO30=0 GPO31=1
 *
 * Runtime:
 * ========
 * It is able to set Lan bypass enable/disable alone, or design hybrid with
 * watchdog timeout(WDTO#).
 * The IO interface for this function is conjunction with Winbond 83627
 * GPO24 (Pair1), GPO60(Pair2) and WDTO#.
 * Refer to Winbond 83627 datasheet for details.
 * The truth table is defined as below:
 *
 * Below setting is to determine system behavior while watchdog timer expired.
 *
 *          GPO27       System behavior
 * ----------------------------------------------
 *          0           Lan-bypass while watchdog timeout
 *          1           System Reset while watchdog timeout
 *
 * Below setting is to determine lan bypass in runtime mode
 *
 *          Pair        Bypass function     GPIO Pin
 *    ---------------------------------------------------------
 *          1           Enable              GPO24 =1
 *          1           Disable             GPO24 =0
 *          2           Enable              GPO60 =1
 *          2           Disable             GPO60 =0
 *
 *          Note: To sete runtime bypass mode, user need to set off-mode bypass
 *        enabled in order to let function activity.
 *
 *-----------------------------------------------------------------------------




 *-----------------------------------------------------------------------------
 */
/*
 * Device Depend Definition : Winbond 83627UHG
 */

#define INDEX_PORT         0x2E
#define DATA_PORT          0x2F
#define SIO_GPIO_22_BIT  0x04
#define SIO_GPIO_23_BIT  0x08
#define SIO_GPIO_24_BIT  0x10
#define SIO_GPIO_27_BIT  0x80
#define SIO_GPIO_30_BIT  0x01
#define SIO_GPIO_31_BIT  0x02
#define SIO_GPIO_60_BIT  0x01


void enter_w83627_config(void)
{
    outportb(INDEX_PORT, 0x87); // Must Do It Twice
    outportb(INDEX_PORT, 0x87);
    return;
}


void exit_w83627_config(void)
{
    outportb(INDEX_PORT, 0xAA);
    return;
}

unsigned char read_w83627_reg(int LDN, int reg)
{
    unsigned char tmp = 0;


    enter_w83627_config();
    outportb(INDEX_PORT, 0x07); // LDN Register
    outportb(DATA_PORT, LDN); // Select LDNx
    outportb(INDEX_PORT, reg); // Select Register
    tmp = inportb( DATA_PORT ); // Read Register
    exit_w83627_config();
    return tmp;
}


void write_w83627_reg(int LDN, int reg, int value)
{
    enter_w83627_config();
    outportb(INDEX_PORT, 0x07); // LDN Register
    outportb(DATA_PORT, LDN); // Select LDNx
    outportb(INDEX_PORT, reg); // Select Register
    outportb(DATA_PORT, value); // Write Register
    exit_w83627_config();
    return;
}


/*Runtime bypass definitions */
#define RUNTIME_BYPASS_PAIR1_LDN          (9)
#define RUNTIME_BYPASS_PAIR1_REG          (0xe5)
#define RUNTIME_BYPASS_PAIR1_BIT          (SIO_GPIO_24_BIT)
#define RUNTIME_BYPASS_PAIR1_ENABLE       (0)
#define RUNTIME_BYPASS_PAIR1_DISABLE      (SIO_GPIO_24_BIT)

#define RUNTIME_BYPASS_PAIR2_LDN          (8)
#define RUNTIME_BYPASS_PAIR2_REG          (0xe5)
#define RUNTIME_BYPASS_PAIR2_BIT          (SIO_GPIO_60_BIT)
```

```
#define RUNTIME_BYPASS_PAIR2_ENABLE      (0)
#define RUNTIME_BYPASS_PAIR2_DISABLE     (SIO_GPIO_60_BIT)

/*Offmode bypass definitions */
#define OFFMODE_BYPASS_PAIR1_LDN         (9)
#define OFFMODE_BYPASS_PAIR1_REG         (0xe5)
#define OFFMODE_BYPASS_PAIR1_BIT         (SIO_GPIO_22_BIT|SIO_GPIO_23_BIT)
#define OFFMODE_BYPASS_PAIR1_ENABLE      SIO_GPIO_22_BIT
#define OFFMODE_BYPASS_PAIR1_DISABLE     SIO_GPIO_23_BIT

#define OFFMODE_BYPASS_PAIR2_LDN         (7)
#define OFFMODE_BYPASS_PAIR2_REG         (0xe1)
#define OFFMODE_BYPASS_PAIR2_BIT         (SIO_GPIO_30_BIT|SIO_GPIO_31_BIT)
#define OFFMODE_BYPASS_PAIR2_ENABLE      SIO_GPIO_30_BIT
#define OFFMODE_BYPASS_PAIR2_DISABLE     SIO_GPIO_31_BIT



void start_watchdog_timer(int watchdog_time)
{
        unsigned char tmp;

        /* clear timeout value */
        write_w83627_reg(0x08, 0xf6, 0x00);

        /* set to count with second */
        tmp=read_w83627_reg(0x08, 0xF5);
        tmp &= ~(0x08);
        write_w83627_reg(0x08, 0xF5, tmp);

        /* clear status bit */
        tmp=read_w83627_reg(0x08, 0xf7);
        tmp &= ~(0x10);
        write_w83627_reg(0x08, 0xf7, tmp);

        /* set WDT Reset Event */
        tmp=read_w83627_reg(0x08, 0xF7);
        tmp = (0x00);
        write_w83627_reg(0x08, 0xF7, tmp);

        /* Set function enable */
        write_w83627_reg(0x08, 0x30, 1);

        /* fill in timeout value */
        write_w83627_reg(0x08, 0xf6, watchdog_time);

        return;
}

void stop_watchdog_timer(void)
{

        /* stop timer */
        write_w83627_reg(0x08, 0xf6, 0);
}

int wd_gpio_init(void)
{
        unsigned char tmp;
        int ret=0;

        /* Set W83627 multiplex pin to WDTO function */
        tmp=read_w83627_reg(0x00, 0x2b);
        tmp &= ~(0x0c);
        tmp |= 0x04;
        write_w83627_reg(0x00, 0x2b, tmp);

        /* clear timeout value */
        write_w83627_reg(0x08, 0xf6, 0x00);

        /* Enable LDN8 watchdog function */
        tmp=read_w83627_reg(0x08, 0x30);
        tmp |= 1;
        write_w83627_reg(0x08, 0x30, tmp);

        /* active GPIO2 group */
        tmp=read_w83627_reg(0x09, 0x30);
        tmp |= 2;
        write_w83627_reg(0x09, 0x30, tmp);

        /* Set GPIO22, 23, 24 and 27 to output mode */
        tmp=read_w83627_reg(0x09, 0xe4);
        tmp &= ~(SIO_GPIO_22_BIT+SIO_GPIO_23_
BIT+SIO_GPIO_24_BIT+SIO_GPIO_27_BIT) ;
        write_w83627_reg(0x09, 0xe4, tmp);

        /* active GPIO3 group */
        tmp=read_w83627_reg(0x07, 0x30);
        tmp |= 1;
        write_w83627_reg(0x07, 0x30, tmp);

        /* Set GPIO30 and 31 to output mode */
        tmp=read_w83627_reg(0x07, 0xe0);
        tmp &= ~(SIO_GPIO_30_BIT + SIO_GPIO_31_BIT) ;
        write_w83627_reg(0x07, 0xe0, tmp);

        /* active GPIO6 group */
        tmp=read_w83627_reg(0x08, 0x30);
        tmp |= 0x4;
        write_w83627_reg(0x08, 0x30, tmp);

        /* Set GPIO60 to output mode */
        tmp=read_w83627_reg(0x08, 0xe4);
        tmp &= ~(SIO_GPIO_60_BIT) ;
        write_w83627_reg(0x08, 0xe4, tmp);
        return ret;

}

void set_bypass_enable_when_system_off(unsigned long
pair_no)
{

        int reg_no, ldn_no;
        unsigned char bit_mask;
        unsigned char en_data;
        unsigned char tmp;

        reg_no=ldn_no=bit_mask=en_data=tmp=0;
        switch(pair_no) {
                case BYPASS_PAIR_1:
                        ldn_no = OFFMODE_BYPASS_
PAIR1_LDN;
                        reg_no = OFFMODE_BYPASS_
PAIR1_REG;
                        bit_mask = OFFMODE_BYPASS_
PAIR1_BIT;
                        en_data = OFFMODE_BYPASS_
PAIR1_ENABLE;
                        break;
                case BYPASS_PAIR_2:
                        ldn_no = OFFMODE_BYPASS_
PAIR2_LDN;
                        reg_no = OFFMODE_BYPASS_
PAIR2_REG;
                        bit_mask = OFFMODE_BYPASS_
PAIR2_BIT;
                        en_data = OFFMODE_BYPASS_
PAIR2_ENABLE;
                        break;
                default:
                        /*un-support pair no, return */
                        return;
        }
        tmp=read_w83627_reg(ldn_no, reg_no);
        tmp &= ~(bit_mask) ;
        tmp |= en_data;
        write_w83627_reg(ldn_no, reg_no, tmp);

        return;
}
```

```
void  set_bypass_disable_when_system_off(unsigned  long pair_no)
{

        int reg_no, ldn_no;
        unsigned char bit_mask;
        unsigned char en_data;
        unsigned char tmp;

        reg_no=ldn_no=bit_mask=en_data=tmp=0;
        switch(pair_no) {
                case BYPASS_PAIR_1:
                        ldn_no = OFFMODE_BYPASS_PAIR1_LDN;
                        reg_no = OFFMODE_BYPASS_PAIR1_REG;
                        bit_mask = OFFMODE_BYPASS_PAIR1_BIT;
                        en_data = OFFMODE_BYPASS_PAIR1_DISABLE;
                        break;
                case BYPASS_PAIR_2:
                        ldn_no = OFFMODE_BYPASS_PAIR2_LDN;
                        reg_no = OFFMODE_BYPASS_PAIR2_REG;
                        bit_mask = OFFMODE_BYPASS_PAIR2_BIT;
                        en_data = OFFMODE_BYPASS_PAIR2_DISABLE;
                        break;
                default:
                        /*un-support pair no, return */
                        return;
        }
        tmp=read_w83627_reg(ldn_no, reg_no);
        tmp &= ~(bit_mask) ;
        tmp |= en_data;
        write_w83627_reg(ldn_no, reg_no, tmp);

        return;
```

```
}

void set_runtime_bypass_enable(unsigned long pair_no)
{
        int reg_no, ldn_no;
    unsigned char tmp, bit_mask, en_data;

        reg_no=ldn_no=bit_mask=en_data=tmp=0;
/*    Note: To sete runtime bypass mode, user need to set off-mode bypass
 *      enabled in order to let function activity.
 */
    set_bypass_enable_when_system_off(pair_no);

    switch(pair_no) {
        case BYPASS_PAIR_1:
                        ldn_no = RUNTIME_BYPASS_PAIR1_LDN;
                        reg_no = RUNTIME_BYPASS_PAIR1_REG;
          bit_mask = RUNTIME_BYPASS_PAIR1_BIT;
          en_data = RUNTIME_BYPASS_PAIR1_ENABLE;
          break;
       case BYPASS_PAIR_2:
                        ldn_no = RUNTIME_BYPASS_PAIR2_LDN;
                        reg_no = RUNTIME_BYPASS_PAIR2_REG;
          bit_mask = RUNTIME_BYPASS_PAIR2_BIT;
          en_data = RUNTIME_BYPASS_PAIR2_ENABLE;
          break;
              default:
                        /*un-support pair no, return */
                        return;

    }
        tmp=read_w83627_reg(ldn_no, reg_no);
        tmp &= ~(bit_mask) ;
        tmp |= en_data;
        write_w83627_reg(ldn_no, reg_no, tmp);
    return;
}

void set_runtime_bypass_disable(unsigned long pair_no)
{

    int reg_no, ldn_no;
    unsigned char tmp, bit_mask, en_data;

    reg_no=ldn_no=tmp=bit_mask=en_data=0;

    switch(pair_no) {
     case BYPASS_PAIR_1:
        ldn_no = RUNTIME_BYPASS_PAIR1_LDN;
        reg_no = RUNTIME_BYPASS_PAIR1_REG;
        bit_mask = RUNTIME_BYPASS_PAIR1_BIT;
        en_data    =    RUNTIME_BYPASS_PAIR1_
DISABLE;
        break;
     case BYPASS_PAIR_2:
        ldn_no = RUNTIME_BYPASS_PAIR2_LDN;
        reg_no = RUNTIME_BYPASS_PAIR2_REG;
        bit_mask = RUNTIME_BYPASS_PAIR2_BIT;
        en_data    =    RUNTIME_BYPASS_PAIR2_
DISABLE;
        break;
  }
    tmp=read_w83627_reg(ldn_no, reg_no);
    tmp &= ~(bit_mask) ;
    tmp |= en_data;
    write_w83627_reg(ldn_no, reg_no, tmp);

    return;
}

void set_wdto_state_system_reset(void)
{
    unsigned char tmp;
```

```
    /* set GPIO27=1 for reset mode */
    tmp=read_w83627_reg(0x9, 0xe5);
    tmp |= SIO_GPIO_27_BIT;
    write_w83627_reg(0x9, 0xe5, tmp);

    return;
}

void set_wdto_state_system_bypass(void)
{
    unsigned char tmp;

    /* set GPIO27=0 for bypass mode */
    tmp=read_w83627_reg(0x9, 0xe5);
    tmp &= ~SIO_GPIO_27_BIT;
    write_w83627_reg(0x9, 0xe5, tmp);

    return;
}

#endif

int main (int argc, char* argv[])
{
    try
    {
        int num = sizeof (id2fun) / sizeof (id2fun[0])
;   //Total function number

        //No parameter. Print the help message
        if (argc < 2)
            RETMSG (-1, PARAMETER_HELP) ;

        //Find and call the coresponding function
        for (int i = 0 ; i < num ; i++)
            if (stricmp (argv[1], id2fun[i].szID)
== 0)
                returnid2fun[i].function
(argc, argv) ;


                RETMSG (-1, “Unknown function name\n”)
;
        }
        catch (char *str)
        {
                printf (“\n%s\n”, str) ;
        }
        catch (...)
        {
                printf (“\nUnknown Exception\n”) ;
        }


        return -1 ;
}
```

Appendix B: Digital Input/Output Control on the GPIO port

The Digital I/O port (DIO) is designed to provide the input and output operations for the system. For sample DIO code, see DIO folder under Driver and Utility on the *Driver and Manual CD*.

Executing the commands through the Command Line:

```
# dio_tst
```

The program will drive output pin with specific value and read status of input pin. If you have external loopback which connects input to output pins directly, the input value should be identical with the output value.

Note:

1. For DOS environment, use DJGPP as compiler and the makefile: Makefile.dos.
2. For Linux, support kernel versions are 2.4.x and 2.6.x. Use the makefile:Makefile.linux.
3. For FreeBSD, support version is FreeBSD 8.0. use the makefile: Makefile.

Build

To build program source code on Linux platform, use the following steps as a guideline:

1. Copy the proper makefile from the Driver and Manual CD to your system
2. Set the access mode with these two parameters by editing the Makefile.linux directly: DIRECT_IO_ACCESS= [0|1] (enter either 1 or 0) and LANNER_DRIVER= [0|1] (enter either 1 or 0). 1 is for direct access and no driver is needed. You will only need to execute the program directly. However, when it equaled to 0, driver installation is needed. Refer to the following Install section for more details.
3. Type make to build source code:

   ```
   make Makefile (Note: omit the file extensions)
   ```

After compiled, the executable program (bpwd_tst) and the driver (bpwd_drv.ko) will be in the bin subdirecto

Install

The installation procedures depend on the access mode that you have set by using the above mentioned method.

If you have set DIRECT_IO_ACCESS=1, driver installation is not necessary. Proceed to the next section on executing

If you have set DIRECT_IO_ACCESS=0, Lanner bypass driver needs to be installed. Install the driver and create a node in the /dev directory as shown in the following example:

For Linux:

Insert module and create node in /dev as below example:

```
#insmod dio_drv.[k]o

#mknod /dev/dio_drv c 240 0
```

For FreeBSD:

Insert module as below example:

```
#kldload -v ./dio_drv.ko
```

I/O Address

DIO Address LDN8	
Address	Description
0x2e	SUPERIO_INDEX
0x2f	SUPERIO_DATA
0x07	BANK_REG
0xE6 (Bit 3)	GPO63 0: Low 1: High
0xE6 (Bit 2)	GPO62 0: Low 1: High
0xE6 (Bit 1)	GPO61 0: Low 1: High
0xE6 (Bit 0)	GPO60 0: Low 1: High

DIO Address LDN9	
Address	Description
0x2e	SUPERIO_INDEX
0x2f	SUPERIO_DATA
0x07	BANK_REG
0xE6 (Bit 3)	GPI24 0: Low 1: High
0xE6 (Bit 2)	GPI25 0: Low 1: High
0xE6 (Bit 1)	GPI26 0: Low 1: High
0xE6 (Bit 0)	GPI27 0: Low 1:High

For example

1. Setting GPO 60-63 all low.

outportb(0x2e, 0x07); LDN8

outportb(0x2f, 0x08);

outportb(0x2e, 0x30); Setting GPIO6.

outportb(0x2f, 0x04);

outportb(0x2e, 0xE4); GP0 60-63

outportb(0x2f, 0x?0); ?:GP0 64-67 Unuse.

outportb(0x2e, 0xE6); GP0 60-63 Uninvert

outportb(0x2f, 0x?0); ?:GP0 64-67 Unuse.

outportb(0x2e, 0xE5); GP0 60-63 1:high

outportb(0x2f, 0x?0); 0:low

2. Setting GPI 24-27.

outportb(0x2e, 0x07); LDN9

outportb(0x2f, 0x09);

outportb(0x2e, 0x30); Setting GPIO6.

outportb(0x2f, 0x04);

outportb(0x2e, 0xE4); GPI 24-27

outportb(0x2f, 0x?F); ?:GPI 20-23 Unuse.

outportb(0x2e, 0xE6); GPI 24-27 Uninvert

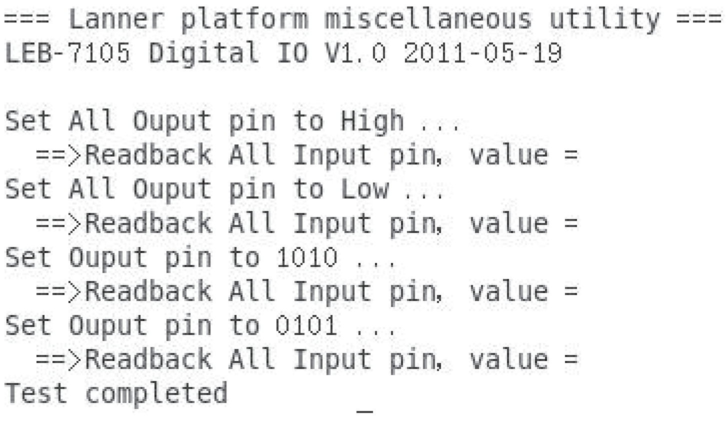
outportb(0x2f, 0x?0); ?:GPI 20-23 Unuse.

Execute

Once build completed, application (and driver) is available in bin sub-directory.

Just run "dio_tst" for Digital IO test. This program will drive output pin with specific value and read status of input pin. If you have external loopback which connects input to output pins directly, the input value should be identical with output value.

screen capture of the execution result:



```
=== Lanner platform miscellaneous utility ===
LEB-7105 Digital IO V1.0 2011-05-19

Set All Ouput pin to High ...
  ==>Readback All Input pin, value =
Set All Ouput pin to Low ...
  ==>Readback All Input pin, value =
Set Ouput pin to 1010 ...
  ==>Readback All Input pin, value =
Set Ouput pin to 0101 ...
  ==>Readback All Input pin, value =
Test completed
                    _
```

Note: For more details, refer to the README file contained within the program

A sample DIO program in C:

```
/**********************************************************
***********************


  ioaccess.c: IO access code for Lanner Platfomr Digital IO
program


  Lanner Platform Miscellaneous Utility
  Copyright(c) 2010- 2011 Lanner Electronics Inc.
  All rights reserved.
*******/



#include "../include/config.h"



#ifdef DJGPP

/* standard include file */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/* For DOS DJGPP */
#include <dos.h>
#include <inlines/pc.h>

#else //DJGPP
/* For Linux */



#ifdef DIRECT_IO_ACCESS
/* For Linux direct io access code */
/* standard include file */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#if defined(LINUX_ENV)
#include <sys/io.h>
#endif

#if defined(FreeBSD_ENV)
#include <machine/cpufunc.h>
#endif



#include <time.h>
#include <stdint.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#define delay(x) usleep(x)
#endif



#ifdef MODULE

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <asm/io.h>
#include <linux/delay.h>

#undef delay
#define delay(x) mdelay(x)
#undef fprintf
#define fprintf(S, A)  printk(A)

#endif //MODULE



#ifdef KLD_MODULE

#include <sys/types.h>
#include <sys/param.h>
#include <sys/systm.h>
```

```
#include <sys/malloc.h>
#include <sys/kernel.h>
#include <sys/bus.h>
#include <sys/errno.h>

#include <machine/bus.h>
#include <machine/resource.h>

#endif

#endif

/* local include file */
#include "../include/ioaccess.h"

#if (defined(MODULE) || defined(DIRECT_IO_ACCESS) || defined(KLD_MODULE))

/*
 *------------------------------------------------------------------------------
 * LEB-7105 Version V1.0
 *
 * The IO interface for Digital DIO is connected to Winbond SIO 83627.
 * Platform provide 4 digital input and 4 digital output.
 * GPIO24-27 as input function, GPIO60-63 as output function
 * Refer to Winbond 83627 datasheet for details.
 * The truth table is defined as below:
 * DIO GPIO pins as follows:
 *              IN          OUT
 * DIO          GP24        GP60
 * DIO          GP25        GP61
 * DIO          GP26        GP62
 * DIO          GP27        GP63
 *-----------------------------------------------------------------------------------
 */
```

```
/*
 * Device Depend Definition : Winbond 83627 SIO
 */
#define INDEX_PORT     0x2E
#define DATA_PORT      0x2F

#define GPIO2X            2
#define GPIO24_BIT        (1 << 4)
#define GPIO25_BIT        (1 << 5)
#define GPIO26_BIT        (1 << 6)
#define GPIO27_BIT        (1 << 7)
#define GPIO_GPIO24_GPIO27_MASK (GPIO24_BIT | GPIO25_BIT | GPIO26_BIT | GPIO27_BIT)

#define GPIO6X            4
#define GPIO60_BIT        (1 << 0)
#define GPIO61_BIT        (1 << 1)
#define GPIO62_BIT        (1 << 2)
#define GPIO63_BIT        (1 << 3)
#define GPIO_GPIO60_GPIO63_MASK (GPIO60_BIT | GPIO61_BIT | GPIO62_BIT | GPIO63_BIT)


void enter_w83627_config(void)
{
    outportb(INDEX_PORT, 0x87); // Must Do It Twice
    outportb(INDEX_PORT, 0x87);
    return;
}

void exit_w83627_config(void)
{
    outportb(INDEX_PORT, 0xAA);
    return;
}

unsigned char read_w83627_reg(int LDN, int reg)
{
    unsigned char tmp = 0;
```

```
        enter_w83627_config();
        outportb(INDEX_PORT, 0x07); // LDN Register
        outportb(DATA_PORT, LDN); // Select LDNx
        outportb(INDEX_PORT, reg); // Select Register
        tmp = inportb( DATA_PORT); // Read Register
        exit_w83627_config();
        return tmp;
}


void write_w83627_reg(int LDN, int reg, int value)
{
        enter_w83627_config();
        outportb(INDEX_PORT, 0x07); // LDN Register
        outportb(DATA_PORT, LDN); // Select LDNx
        outportb(INDEX_PORT, reg); // Select Register
        outportb(DATA_PORT, value); // Write Register
        exit_w83627_config();
        return;
}

void dio_gpio_init(void)
{
        unsigned char tmp;
                /* Enable GPIO 6x function */
                tmp=read_w83627_reg(0x08, 0x30);
                tmp |= GPIO6X;
                write_w83627_reg(0x08, 0x30, tmp);

                /* set GPIO60~63 as Output function */
                tmp=read_w83627_reg(0x08, 0xE4);
                tmp &= ~(GPIO_GPIO60_GPIO63_MASK);
                write_w83627_reg(0x08, 0xE4, tmp);

                /* set GPIO60~63 as uninvert */
                tmp=read_w83627_reg(0x08, 0xE6);
                tmp &= ~(GPIO_GPIO60_GPIO63_MASK);
                write_w83627_reg(0x08, 0xE6, tmp);

                /* set GPIO60~63 generate high signal */
                tmp=read_w83627_reg(0x08, 0xE5);
                tmp |= GPIO_GPIO60_GPIO63_MASK;
                write_w83627_reg(0x08, 0xE5, tmp);

                /* Enable GPIO 2x function */
                tmp=read_w83627_reg(0x09, 0x30);
                tmp |= GPIO2X;
                write_w83627_reg(0x09, 0x30, tmp);

                /* set GPIO24~27 as Input function */
                tmp=read_w83627_reg(0x09, 0xE4);
                tmp |= GPIO_GPIO24_GPIO27_MASK;
                write_w83627_reg(0x09, 0xE4, tmp);

        /* set GPIO24~27 as uninvert */
                tmp=read_w83627_reg(0x09, 0xE6);
                tmp &= ~(GPIO_GPIO24_GPIO27_MASK);
                write_w83627_reg(0x09, 0xE6, tmp);

        return;
}
void dio_set_output(unsigned char out_value)
{
                unsigned char tmp;

                tmp = read_w83627_reg(0x08,0xE5);
                tmp &= ~GPIO_GPIO60_GPIO63_MASK;
                tmp |= out_value;
                write_w83627_reg(0x08, 0xE5, tmp);
                delay(333);
        return;
}

unsigned char dio_get_input(void)
{
        unsigned char tmp;

        tmp=read_w83627_reg(0x09, 0xE5);
        tmp &= GPIO_GPIO24_GPIO27_MASK;
        return tmp;
}

#endif
```

Appendix C: Terms and Conditions

Warranty Policy

1. All products are under warranty against defects in materials and workmanship for a period of one year from the date of purchase.
2. The buyer will bear the return freight charges for goods returned for repair within the warranty period; whereas the manufacturer will bear the after service freight charges for goods returned to the user.
3. The buyer will pay for repair (for replaced components plus service time) and transportation charges (both ways) for items after the expiration of the warranty period.
4. If the RMA Service Request Form does not meet the stated requirement as listed on "RMA Service," RMA goods will be returned at customer's expense.
5. The following conditions are excluded from this warranty:

Improper or inadequate maintenance by the customer Unauthorized modification, misuse, or reversed engineering of the product Operation outside of the environmental specifications for the product.

RMA Service

Requesting a RMA#

6. To obtain a RMA number, simply fill out and fax the "RMA Request Form" to your supplier.
7. The customer is required to fill out the problem code as listed. If your problem is not among the codes listed, please write the symptom description in the remarks box.
8. Ship the defective unit(s) on freight prepaid terms. Use the original packing materials when possible.
9. Mark the RMA# clearly on the box.

Note: Customer is responsible for shipping damage(s) resulting from inadequate/loose packing of the defective unit(s). All RMA# are valid for 30 days only; RMA goods received after the effective RMA# period will be rejected.

Appendix C

RMA Service Request Form

When requesting RMA service, please fill out the following form. Without this form enclosed, your RMA cannot be processed.

RMA No:	Reasons to Return: □ Repair(Please include failure details) □ Testing Purpose
Company:	Contact Person:
Phone No.	Purchased Date:
Fax No.:	Applied Date:

Return Shipping Address: ____

Shipping by: □ Air Freight □ Sea □ Express____________

□ Others:_________________

Item	Model Name	Serial Number	Configuration

Item	Problem Code	Failure Status

*Problem Code:

01:D.O.A.
02: Second Time R.M.A.
03: CMOS Data Lost
04: FDC Fail
05: HDC Fail
06: Bad Slot

07: BIOS Problem
08: Keyboard Controller Fail
09: Cache RMA Problem
10: Memory Socket Bad
11: Hang Up Software
12: Out Look Damage

13: SCSI
14: LPT Port
15: PS2
16: LAN
17: COM Port
18: Watchdog Timer

19: DIO
20: Buzzer
21: Shut Down
22: Panel Fail
23: CRT Fail
24: Others (Pls specify)

Request Party

____

Authorized Signature / Date

Confirmed By Supplier

____

Authorized Signature / Date